

DAM
Desarrollo de Aplicaciones Multiplataforma
2º Curso

AD
Acceso a Datos

UD 8
Programación de componentes
de acceso a datos
Parte 2

IES BALMIS
Dpto Informática
Curso 2019-2020
Versión 2 (01/2020)

7. Programación de aplicaciones en Servidores Web

UD8Ejer701 - JavaToJSP

Tenemos una aplicación denominada **JavatoJSP** en Java que muestra una tabla con 100 letras mayúsculas generadas de forma aleatoria.

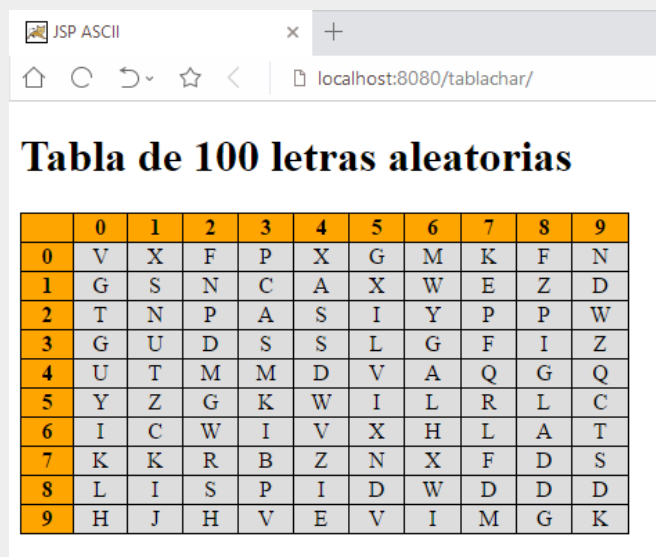
```
+---+---+---+---+---+---+---+---+---+---+---+
|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
+---+---+---+---+---+---+---+---+---+---+---+
| 0 | N | F | F | Z | O | C | Y | N | F | U |
+---+---+---+---+---+---+---+---+---+---+---+
| 1 | T | M | A | T | L | L | V | R | R | B |
+---+---+---+---+---+---+---+---+---+---+---+
| 2 | T | W | V | N | E | Z | Q | L | Y | W |
+---+---+---+---+---+---+---+---+---+---+---+
| 3 | L | N | I | L | V | U | U | W | C | N |
+---+---+---+---+---+---+---+---+---+---+---+
| 4 | N | Y | C | P | Q | Y | I | Q | F | E |
+---+---+---+---+---+---+---+---+---+---+---+
| 5 | N | X | Z | Q | R | E | E | W | K | N |
+---+---+---+---+---+---+---+---+---+---+---+
| 6 | H | A | R | O | U | B | Z | U | T | O |
+---+---+---+---+---+---+---+---+---+---+---+
| 7 | G | M | P | E | G | Q | Y | B | X | X |
+---+---+---+---+---+---+---+---+---+---+---+
| 8 | H | J | A | C | M | S | Z | N | O | D |
+---+---+---+---+---+---+---+---+---+---+---+
| 9 | J | D | A | N | T | B | V | M | G | X |
+---+---+---+---+---+---+---+---+---+---+---
```

Para ello realizamos dos procesos:

1. Cargar una matriz 10x10 de char y rellenarla generando números aleatorios entre el 65 y el 90 y obtenemos su carácter ASCII
2. Mostramos la tabla realizando dos bucles anidados

Deseamos crear la misma aplicación pero de tipo Web, siendo la salida un archivo HTML generado desde una página JSP.

La salida será:



	0	1	2	3	4	5	6	7	8	9
0	V	X	F	P	X	G	M	K	F	N
1	G	S	N	C	A	X	W	E	Z	D
2	T	N	P	A	S	I	Y	P	P	W
3	G	U	D	S	S	L	G	F	I	Z
4	U	T	M	M	D	V	A	Q	G	Q
5	Y	Z	G	K	W	I	L	R	L	C
6	I	C	W	I	V	X	H	L	A	T
7	K	K	R	B	Z	N	X	F	D	S
8	L	I	S	P	I	D	W	D	D	D
9	H	J	H	V	E	V	I	M	G	K

Para mostrar la tabla de esta forma necesitaremos utilizar CSS:

```
<style>
    table {
        border-collapse: collapse;
        border: 1px solid black;
        background-color: #dddddd;
        width: 440px;
        text-align: center;
    }
    td {
        width: 40px;
    }
    .cabecera {
        background-color: orange;
        font-weight: bold;
    }
</style>
```

El código JSP a completar sería:

```
<h1>Tabla de 100 letras aleatorias</h1>
<%
char[][] matriz = new char[10][10];

// Carga de datos
for (int i=0; i<10; i++) {
    for (int j=0; j<10; j++) {
        // Número entre el 65 y 90 -> 27 números posibles
        int randomNum = ThreadLocalRandom.current().nextInt(65, 91);
        matriz[i][j]=(char) randomNum;
    }
}

// Mostrar matriz
%>
<table border="1">
    <tr>
        <td class="cabecera"></td>
        <% for (int i=0; i<10; i++) { %>
            <td class="cabecera"><%= i %></td>
        <% } %>
    </tr>
    <tr>
        <td class="cabecera">0</td>
        <td>S</td>
        <td>J</td>
        <td>G</td>
        <td>K</td>
        <td>L</td>
        <td>B</td>
        <td>V</td>
        <td>C</td>
        <td>X</td>
        <td>Q</td>
    </tr>
</table>
```

Crea el proyecto **JavaToJSPhtml** de tipo **"Java Web => Web Application"** con **Apache Tomcat** con el **Context Path = "/tablachar"** y completa un **index.jsp** para obtener el resultado deseado.

Publica desde NetBeans en el servidor de Apache Tomcat, prueba su funcionamiento. Captura la pantalla del navegador con el resultado **poniendo como title tu nombre**.

Crea el WAR de tu proyecto para su distribución.

ENTREGAR:

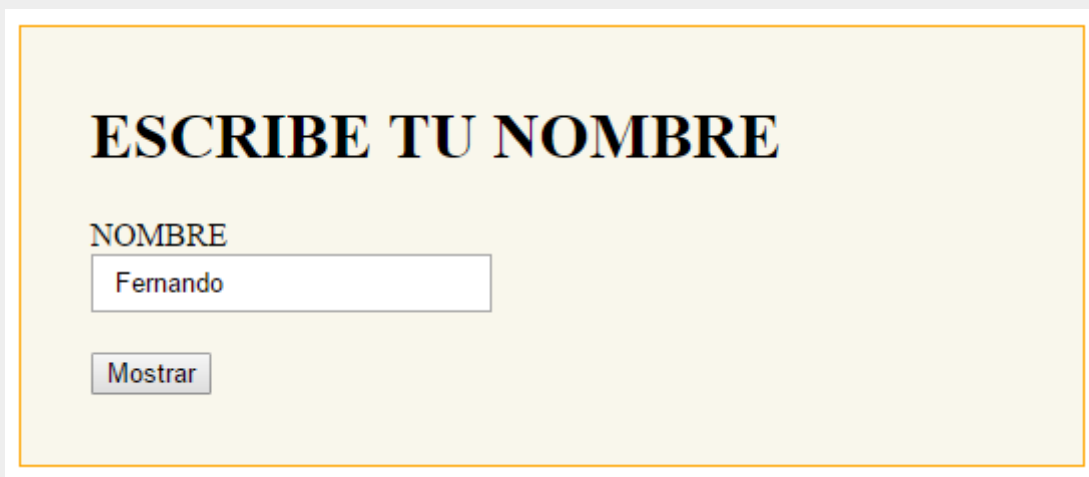
JPG => Captura de pantalla del navegador

ZIP => Proyecto en NetBeans

WAR => Archivo empaquetado del proyecto

UD8Ejer702 - JSPNombre

Completa el proyecto denominado **JSPNombre** de tipo **"Java Web => Web Application"** con **Apache Tomcat** con el **Context Path = "/jspnombre"** que muestre un formulario pidiendo tu nombre y al pulsar sobre el botón submit, muestre un HTML (utilizando JSP) con el contenido del nombre en mayúsculas y con un espacio separando cada letra.

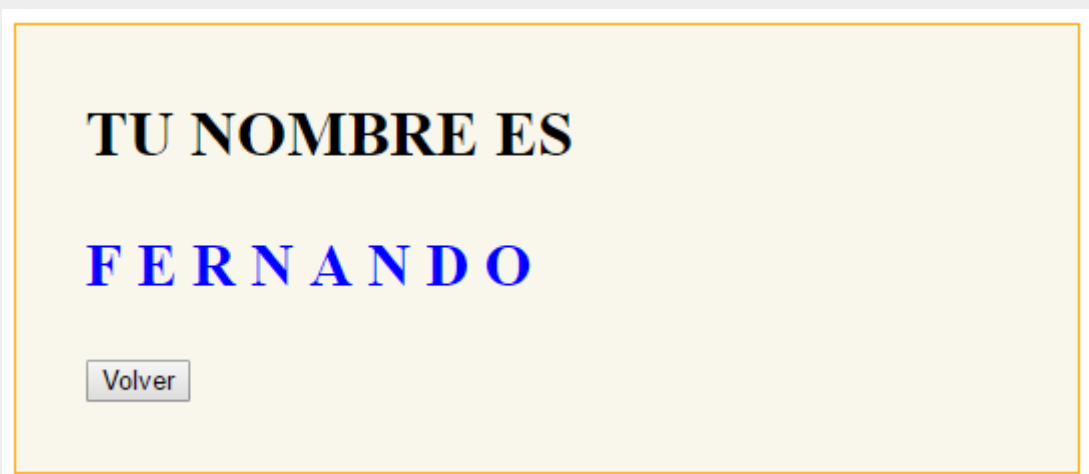


ESCRIBE TU NOMBRE

NOMBRE

Mostrar

Y al pulsar obtendríamos:



TU NOMBRE ES

FERNANDO

Volver

8. Creación de aplicaciones web

UD8Ejer801 - GimMongo

Crear una Aplicación Web que pida el dni de un cliente y muestre sus datos.

Para ello, deberá acceder a MongoDB y buscar la información en la colección **clientes** de la BD **gimnasio** de MongoDB.

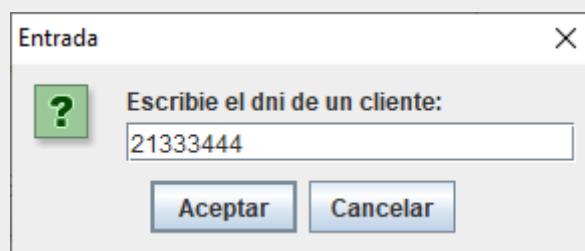
Para completar el proceso de diseño, crearemos varios proyectos:

GimMongoPruebas: Será una aplicación de tipo **"Java => Java Application"** que contendrá dos clases

1. **Cliente:** clase serializable para almacenar la información obtenida de MongoDB
2. **Gimnasio:** clase con los métodos de acceso a MongoDB. Tendrá un método **"Cliente obtieneCliente(String dni)"** que conectará con MongoDB y devolverá un objeto de la clase Cliente con los datos obtenidos.
3. Habrá una clase del proyecto que contendrá el main y que pedirá el dni utilizando Swing y mostrará los datos del cliente usando toString.

GimMongoClass: Será un proyecto de tipo **"Java => Java Class Library"** con el contenido de las clases Cliente y Gimnasio del proyecto anterior, y que estructuraremos para importar desde **com.dam.gimmongoclass**

AppGimMongo: Será un proyecto de tipo **"Java => Java Application"** que usando el componente GimMongoClass pedirá un dni utilizando Swing y mostrará el resultado con toString.



```
Salida - GimMongoPruebas (run) X
run:
Conectando a MongoDB.
Seleccionando BD y colección.
En la BD ..... 'gimnasio'
En la Colección .. 'gimnasio.clientes'
El número de documentos es 4
Desconectando de MongoDB.
Cliente{dni=21333444, nombre=Gloria, apellidos=Rodríguez, fecha_nac=1998-12-05, telefono=651322478}
BUILD SUCCESSFUL (total time: 31 seconds)
```

JspGimMongo: Será un proyecto de tipo "**Java Web => Web Application**" con el servidor **Apache Tomcat** que usando el componente GimMongoClass pedirá un dni con un formulario HTML y al pulsar en el botón mostrará el resultado con una tabla de HTML.

BUSCAR CLIENTE

DNI

CLIENTE

DNI: **21333444**

NOMBRE: **Gloria**

APELLIDOS: **Rodríguez**

NACIMIENTO: **1998-12-05**

TELÉFONO: **651322478**

UD8Ejer802 - JspTableLibros

Crear una **Aplicación Web** que cargue los datos de un libro en un objeto de la clase Libro y muestre sus datos.

JspTableLibros: Será una aplicación de tipo **"Java => Java Application"** con el servidor **Apache Tomcat** que cargue un **ArrayList<Libros>** y muestre sus datos.

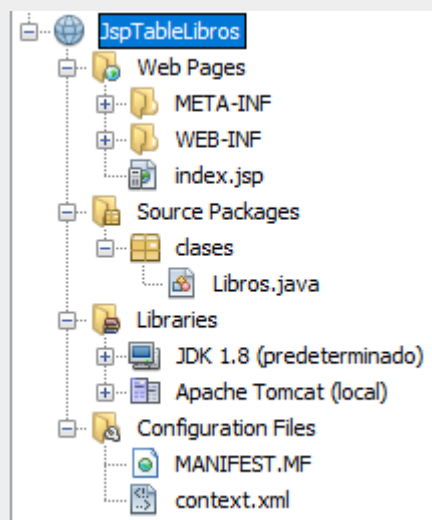
Se deberá crear la clase Libros:

```
public class Libros implements Serializable {
    private int id;
    private String titulo;
    private String autor;
    ...
}
```

El Context Path será:

Server:	Apache Tomcat (local)
Java EE Version:	Java EE 6 Web
Context Path:	/jsptablelibros

La estructura será:



El código del JSP para cargar los datos será:

```
<%
    ArrayList<Libros> datos = new ArrayList<Libros>();

    Libros registro;

    registro = new Libros();
    registro.setId(1);
    registro.setTitulo("Macbeth");
    registro.setAutor("William Shakespeare");
    datos.add(registro);

    registro = new Libros();
    registro.setId(2);
    registro.setTitulo("La Celestina (Tragicomedia de Calisto y Melibea)");
    registro.setAutor("Fernando de Rojas");
    datos.add(registro);
%>
```

```
registro = new Libros();
registro.setId(3);
registro.setTitulo("El Lazarillo de Tormes");
registro.setAutor("Anónimo");
datos.add(registro);

registro = new Libros();
registro.setId(4);
registro.setTitulo("20.000 Leguas de Viaje Submarino");
registro.setAutor("Julio Verne");
datos.add(registro);

registro = new Libros();
registro.setId(5);
registro.setTitulo("Alicia en el País de las Maravillas");
registro.setAutor("Lewis Carrol");
datos.add(registro);

registro = new Libros();
registro.setId(6);
registro.setTitulo("Cien Años de Soledad");
registro.setAutor("Gabriel García Márquez");
datos.add(registro);

registro = new Libros();
registro.setId(7);
registro.setTitulo("La tempestad");
registro.setAutor("William Shakespeare");
datos.add(registro);
%>
```

La salida será:

Número de registros: 7

ID LIBRO	TÍTULO	AUTOR
1	Macbeth	William Shakespeare
2	La Celestina (Tragicomedia de Calisto y Melibea)	Fernando de Rojas
3	El Lazarillo de Tormes	Anónimo
4	20.000 Leguas de Viaje Submarino	Julio Verne
5	Alicia en el País de las Maravillas	Lewis Carrol
6	Cien Años de Soledad	Gabriel García Márquez
7	La tempestad	William Shakespeare

UD8Ejer803 - JspLibro

Crear una **Aplicación Web** que cargue los datos de un libro en un objeto de la clase Libro y muestre sus datos.

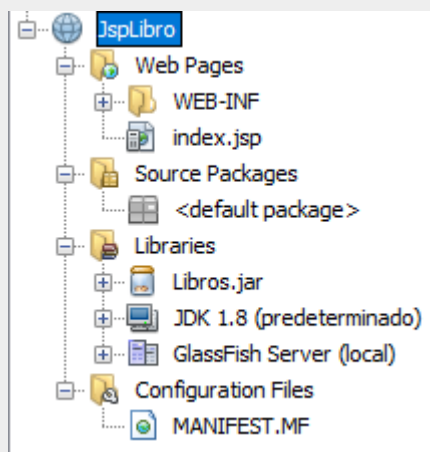
JspLibro: Será una aplicación de tipo "**Java => Java Application**" con el servidor **GlassFish** que usará el componente denominado Libros con la clase Serializable Libros empaquetada en el ejercicio 401 importable desde **com.dam.bibliotecah.libros**:

```
public class Libros implements Serializable {
    private int id;
    private String titulo;
    private String autor;
    ...
}
```

El Context Path será:

Server:	GlassFish Server (local)
Java EE Version:	Java EE 7 Web
Context Path:	/jsplibro

La estructura será:



El código del JSP será:

```
<h1>SESSION ID</h1>
<h2><% out.println(session.getId()); %></h2>

<%
    String titulo = "Datos del Libro";
    Libros libro = new Libros(1, "Macbeth", "William Shakespeare");
%>
<h1><%= titulo %></h1>
<h1><%= libro.toString() %></h1>
```